REMARKS

This Preliminary Amendment cancels without prejudice original claims 1 to 12 in the underlying PCT Application No. PCT/EP00/00386, and adds without prejudice new claims 13 to 26. The new claims conform the claims to U.S. Patent and Trademark Office rules and do not add new matter to the application.

In accordance with 37 C.F.R. § 1.121(b)(3), the Substitute Specification (including the Abstract, but without the claims) contains no new matter. The amendments reflected in the Substitute Specification (including Abstract) are to conform the Specification and Abstract to U.S. Patent and Trademark Office rules or to correct informalities. As required by 37 C.F.R. § 1.121(b)(3)(iii) and § 1.125(b)(2), a Marked Up Version Of The Substitute Specification comparing the Specification of record and the Substitute Specification also accompanies this Preliminary Amendment. In the Marked Up Version, shading indicates added text and bracketing indicates deleted text. Approval and entry of the Substitute Specification (including Abstract) is respectfully requested.

The underlying PCT Application No. PCT/EP00/00386 includes an International Search Report, dated August 1, 2000. The Search Report includes a list of documents that were uncovered in the underlying PCT Application. A copy of the Search Report accompanies this Preliminary Amendment.

The underlying PCT Application No. PCT/EP00/00386 also includes an International Preliminary Examination Report, dated July 30, 2001. An English translation of the International Preliminary Examination Report accompanies this Preliminary Amendment.

Applicants assert that the subject matter of the present application is new, non-obvious, and useful. Prompt consideration and allowance of the application are respectfully requested.

Respectfully Submitted,

KENYON & KENYON

Dated: 8/13/2001

By: Richard Mayer (Reg. No. 33,865 Aaron C. DEDITCH)

Richard L. Mayer
(Reg. No. 22,490)
One Broadway
New York, NY 10004
(212) 425-7200 (telephone)
(212) 425-5288 (facsimile)

CUSTOMER NO. 26646

[
]

METHOD FOR GRAPHICALLY REPRESENTING AND/OR PROCESSING VALUES OF DATA TYPES

[Background of the Invention
]FIELD OF THE INVENTION

The present invention relates to a method for graphically representing and/or processing values of data types of a formally defined data structure existing as a value tree.

BACKGROUND INFORMATION

Programming and description languages may require a formal syntax to describe data types and their values. [Therefore, m]Methods are [known]believed to be available for defining data structures using a formal notation, independently of a specific language. [Generally, t]The data types defined in accordance with one of these methods [can]may be of a[ny] desired complexity and may dynamically change in part. [It is]Thus,[therefore, necessary that] such data types should be clearly represented for display purposes or to enable user input of values. This [is particularly necessary]should be done for the management of communications networks, since very complex data structures [are often]may be processed in these networks.

In [most]various languages and syntax notations, rules[already] exist for textually representing values. Thus, for example, when working with ASN.1 data types in accordance with ITU-T recommendation X.208, the value is represented as a character string in a manner[that], which is not believed to be very straightforward.

[T]It is believed that the structure of the data type [is more easily recognized]may be recognizable when the[known] form of

representation [as]of a tree is used. A representation of this kind is [described]discussed in the [firm publication]reference "IBM TMN Development Platform", Piscataway, N[.J.]ew Jersey, U.S.A., 1998. Another [known]method is believed to include[s] assigning the value of an attribute to a graphic component by manually creating rules, for example, defining the color of the graphic object by the value of the data type, [as described]discussed, for example, in the [company publication, Objective]reference "Objective Systems Integrators: ["]NetExpert Framework Overview", Folsom, [CA]California, U.S.A., 1997.

Since any data type definitions at all [are]may be possible, it is believed that the [known]available methods may expend a great deal of time programming a separate graphical user-interface window for each new data type.

[
The object]

SUMMARY OF THE INVENTION

An exemplary embodiment and/or exemplary method of the present invention is directed to [reduce]reducing th[is]e time expenditure on programming and to[, nevertheless,] achieve a straightforward and clear representation, which will enable the data structure to be tested and, if [necessary, processed.

This objective is achieved by the method in accordance with]desired, processed.

Another exemplary embodiment and/or exemplary method of the present invention [in]is directed to providing that:

- a window [is]may be assigned as a graphical user interface to the data structure;
- generic, scalable, graphical user-interface components [are]may be inserted hierarchically in the window, the value tree of the data structure being mapped onto the user-interface components;

- the graphical user interface components [are]may be in a relation to the nodes of the value tree in a manner that is recognizable to the user; and
- [that]a graphical or textual representation of the value [is]may be selectable for each subtree of the value tree.

The clear, straightforward representation of a complex data type using the exemplary method in accordance with the present invention may save[s] the user from having to search extensively for definitions and [helps him]may help to avoid errors when inputting values for this data type. In this context, the exemplary method of present invention [offers]may provide the[possibility of] concealing of redundant information and [of]the presenting [important]of information in detail, since [in the]an exemplary embodiment and/or exemplary method [according to]of the present invention[,] is directed to providing that each user has a chance to decide which information should be displayed and for which information a compact representation suffices. [In particular, the]Another exemplary embodiment and/or exemplary method [according to]of the present invention [allows]is directed to providing a simple value assignment to be made in the processing of the data types.

[One further refinement]Another exemplary embodiment and/or exemplary method of the present invention [provides]is directed to providing for [an especially]a simple and secure value assignment to be carried out in that, for a processing of the value tree, a list of all values which are compatible with respect to assignment with the represented data type [is]may be derived for each node, and that, in each case, one value [is]may be selected from the list for a value assignment. [In the process, t]To avoid input errors, it [can]may be provided when compiling value lists, that the number of values to be accepted in the list [be]are restricted in accordance with predefined rules, depending on the current

context.

Another [refinement]exemplary embodiment and/or exemplary method of the present invention [provides]is directed to providing for a visualization of the window to be first undertaken at the time of an initialization of the graphical user interface and, after that, data, [in particular]for example, value lists, to be initialized, which [are]may be derived for a processing. As a result of the faster display build-up associated therewith, the user [can]may already obtain an overview of the data structure before all data required for displaying and processing the data type are initialized.

[A]Another exemplary embodiment and/or exemplary method of the present invention is directed to providing that a data type [can also]may be graphically displayed when no additional information, described as metadata, on the data type is available in the graphical interface application, when[, in accordance with another refinement of the present invention,] the value to be represented is transferred in a transfer syntax, which contains all necessary information for the representation with respect to the data type and the value assignment.

[In accordance with one advantageous specific embodiment of the method according to]Another exemplary embodiment and/or exemplary method of the present invention[,] is directed to providing data types, whose exact type assignment [can]may first be determined at the execution time in accordance with the late binding principle, a[re]nd inserted as a dynamically changeable subtree in the value tree represented by the graphical user interface. From this, [one derives the]an advantage for these data types may be that the representation of the current value assignment does not first require opening subwindows[] (or subforms), but [can]may take place directly

in the main window.

Another [advantageous specific embodiment provides that,] exemplary embodiment and/or exemplary method of the present invention is directed to providing that for data types whose exact type assignment [is] may be first defined in accordance with the late binding principle at the execution time by the marking of another node (for example, "ANY DEFINED BY" in the description language ASN.1), the user [is] may be prompted to input whether the assignment should be carried out automatically or following a manual input. Thus, an assignment [is also possible] may occur when the information required for the automatic assignment is not available.

[In accordance with another advantageous refinement, values can] Another exemplary embodiment and/or exemplary method of the present invention is directed to providing that values may be transferred from one subtree into another by intermediately storing and clicking on the subtree in question. In the process, an assignment compatibility of the data types may be assigned to the subtrees[is, in fact, necessary]. However, it [is] may not be necessary for the data types to be instantiated within the same value tree.

Another [advantageous refinement makes possible] exemplary embodiment and/or exemplary method of the present invention is directed to providing a simple use, together with other programs, and the simple mapping of the [general]overall identity in that the exemplary method [is] may be implemented by one or more program modules that [can] may be integrated in the application programs.

[Yet a] Another [advantageous refinement] exemplary embodiment and/or exemplary method of the present invention [provides] is directed to providing that additional information to be displayed [can] may be stored for each node of the value tree

which [can]may be uniquely named by the displayed type and the relation to the higher-order type. In this manner, additional text elements [can]may be displayed for specific data types, which [are]may be produced at any point in time following the program creation, and which [can]may be dynamically integrated in the interface at the execution time.

Another exemplary embodiment and/or exemplary method of the present invention [provides that it be]is directed to providing continual[ly checked] checking during inputting of a value to determine whether the input value is permissible for the corresponding data type, and whether the input value is identical to the currently active value of the data type, and that the result be made known to the user. [The purpose of this embodiment is]Another exemplary embodiment and/or exemplary method of the present invention is directed to further [enhance]enhancing security and the speed attained in the processing of data types.

[In addition,]Another exemplary embodiment and/or exemplary method of the present invention is directed to providing that the display and/or the processing [can]may be facilitated in that the display format [can]may be altered already at the time that a value is input and, thus, for example, a numerical value [is]may be either displayed as a decimal or binary value, before a value is accepted into the value tree.

[

Exemplary embodiments of the present invention are represented by several figures in the drawing and are more closely explained in the following description. The figures show:

Figure 1]

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 shows a window of a graphical user interface [in accordance with the]according to an exemplary embodiment and/or exemplary method of the present invention[;].

Figure 2[an example of the] shows a formal data definition in accordance with ASN.1[; and].

Figure 3[ ] shows the principle of a generic, graphic representation.

[In the form of a structured text,]

DETAILED DESCRIPTION

Figure 1 shows the relevant graphical user interface as a window 1, in which all included data types are contained as graphical user-interface components. The root of the value tree (Figure 3) is indicated by the GetArgument SEQUENCE. Input bars 2, 3 and 4, having the labels "globalForm", "localForm" and "baseManagedObjectInstance" correspond to the leaf objects of the value or GUI tree. Buttons 5 through 9 can be used to select between a textual representation (minus) and a graphic representation (plus) for each node and each leaf object.

Other buttons 10, 11 (radio buttons) may be used to select between two data types to be alternatively processed (in this case, leaf objects). Input bars 2, 3, 4 are each provided with a button 12, 13, 14, which can be used to open a window 15 for selecting values stored as constants.

In addition, the graphical user interface has buttons 16, 17 for terminating the processing and storing the processed data, as well as for exiting the graphical user interface without storing the data.

Figure 2 shows the definitions of the data types "GetArgument" and "BaseManagedObjectClass" in the form of a structured text. The type "GetArgument" includes two components, "baseManagedObjectClass" and "baseManagedObjectInstance", the names being given to the left, and the types of the components to the right, namely "BaseManagedObjectClass" and GraphicString". The type "BaseManagedObjectClass" is a CHOICE

having likewise two components, namely "globalForm" of type "OBJECTIDENTIFIER" and "localForm" of type "INTEGER".

Figure 3 [depicts] shows the principle of a generic, graphic representation of the formal data definition illustrated in Figure 2. In this context, the value tree is shown to the left, while the graphic representation is indicated to the right. In this instance and in the text that follows, the abbreviation GUI (= Graphical User Interface) is also used for the graphical user interface. Solid lines running between the blocks signify that the underlying or subjacent elements are contained in the node situated above them, while the dotted lines signify that each graphic representation is in relation to the particular data type.

[

Figure 1 illustrates the relevant graphical user interface as a window 1, in which all included data types are contained as graphical user-interface components. The root of the value tree (Figure 3) is indicated by the GetArgument SEQUENCE. Input bars 2, 3 and 4, having the labels "globalForm", "localForm" and "baseManagedObjectInstance" correspond to the leaf objects of the value or GUI tree. Buttons 5 through 9 can be used to select between a textual representation (minus) and a graphic representation (plus) for each node and each leaf object.

Other buttons 10, 11 (radio buttons) can be used to select between two data types to be alternatively processed (in this case, leaf objects). Input bars 2, 3, 4 are each provided with a button 12, 13, 14, which can be used to open a window 15 for selecting values stored as constants.

In addition, the graphical user interface has buttons 16, 17 for terminating the processing and storing the processed data, as well as for exiting the graphical user interface without storing.

Abstract

In]

ABSTRACT OF THE DISCLOSURE

In a method for graphically representing and/or processing values of data types of a formally defined data structure existing as a value tree, a window is assigned as a graphical user interface to the data structure. Generic, scalable, graphical user-interface components are inserted hierarchically in the window, the value tree of the data structure being mapped onto the user-interface components. The graphical user-interface components are in a relation to the nodes of the value tree that is recognizable to the user. A graphical or textual representation of the value is selectable for each subtree of the value tree.[

]